Predicate (first-order) logic

02 April 2024 08:19

Next week: predicate logic as a language for modeling systems

- $\forall x \forall y$ connected$(x,y) \longrightarrow$ connected(y,x)
  $\forall x$ server$(x) \vee$ client(x)
  $\forall x$ $\neg$server$(x) \vee \neg$client(x)
  $\forall x$ printer$(x) \longrightarrow$ server(x)
  ...

SYNTAX (how we write formulas)
SEMANTICS (attaching meaning to formulas)

- $\forall x\,(x \neq 0 \longrightarrow \exists y\,(x = y \cdot y))$ če kvantificiramo: over $\mathbb{Z}$ or $\mathbb{Q}$ then false; over $\mathbb{R}$ then true

  $\forall x\,(\neg(x=0) \longrightarrow \exists y.\ x \cdot y = 1)$ is true over $\mathbb{Q}$ and false over $\mathbb{Z}$.

Signature $(\mathcal{P}, \mathcal{F})$; $\mathcal{P}$ = set of predicate symbols: Each $P \in \mathcal{P}$ has an associated arity $Ar(P) \geq 0$
↳ specifies the vocabulary
$\mathcal{F}$ = set of function symbols: Each $f \in \mathcal{F}$ ... $Ar(f) \geq 0$.

EX.:
- $\mathcal{P} = \{$ connected$/2$, server$/1$, priner$/1$, client$/1\}$, $\mathcal{F} = \emptyset$
- $\mathcal{P} = \{\geq/2\}$, $\mathcal{F} = \{\cdot/2, 0/0, 1/0\}$

The signature determines rules for generating
terms - express elements of the universe of discourse
formulas - express properties

TERMS:

- variables
- if f is a function symbol of arity n and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.

- 0 is a term (officially the term is $0()$), $x \cdot y$ and $y \cdot y$ are terms (officially $\cdot(x,y)$ and $\cdot(y,y)$)

FORMULAS:

- if $P \in \mathcal{P}$ has arity n and $t_1, \dots, t_n$ are terms, then $P(t_1, \dots, t_n)$ is a formula
- if t_1 and t_2 are terms, then $t_1 = t_2$ is a formula

} atomic formulas

- if ϕ and ψ are formulas, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\phi \rightarrow \psi$, $\top$, $\bot$ are formulas
- if ϕ is a formula and x a variable, then $\forall x\,\phi$ and $\exists x\,\phi$ are formulas.

Structure

02 April 2024 09:19

The semantics to logical formulas is is given by a **STRUCTURE** (model in [HR])

$\exists y\,(x = y \cdot y)$

Let's interpret this in $\mathbb{N}$. (officially in the structure $\mathcal{N} = (\mathbb{N}, \dots)$)

The formula expresses a property of x (x is a perfect square).

x is a **free variable** (not a bound variable)

y is a **bound variable** (in the scope of a quantifier that bounds it)

A formula expresses a relation between the values of its free variables.

A formula with no free variables is called a **sentence**.

A **structure** for a signature $(\mathcal{P}, \mathcal{F})$ is $\mathcal{M} = (M, \{P^{\mathcal{M}}\}_{P \in \mathcal{P}}, \{f^{\mathcal{M}}\}_{f \in \mathcal{F}})$, where

M = underlying set

$P^{\mathcal{M}} \subseteq M^n$ where $n = Ar(P)$

$f^{\mathcal{M}} : M^n \to M$ where $n = Ar(f)$.

EXAMPLES:

- $\mathcal{N}$ the natural interpretation of $(\leq, +, \cdot, 0, 1)$ is $\mathcal{N} = (\mathbb{N}, \leq^{\mathcal{N}}, +^{\mathcal{N}}, \cdot^{\mathcal{N}}, 0^{\mathcal{N}}, 1^{\mathcal{N}})$, where
  $m \leq^{\mathcal{N}} n \iff m \leq n$
  $m +^{\mathcal{N}} n \iff m + n$
  $m \cdot^{\mathcal{N}} n := m \cdot n$
  $0^{\mathcal{N}} := 0$
  $1^{\mathcal{N}} := 1$

- $\mathcal{N}' = (\mathbb{N}, \leq^{\mathcal{N}'}, +^{\mathcal{N}'}, \cdot^{\mathcal{N}'}, 0^{\mathcal{N}'}, 1^{\mathcal{N}'})$
  $m \leq^{\mathcal{N}'} n \iff m > 2^n$
  $m +^{\mathcal{N}'} n := m \cdot n$
  $m \cdot^{\mathcal{N}'} n := \lfloor (\log_2 m) + n \rfloor$
  $0^{\mathcal{N}'} := 17$
  $1^{\mathcal{N}'} := 17$

$t^{\mathcal{M}}_{\rho}$ = the interpretation of term t is structure $\mathcal{M}$ via ρ

ρ is an **environment** ρ: Variables $\to M$

$x^{\rho} := \rho(x)$

$(f(t_1, \dots, t_n))^{\rho} := f^{\mathcal{M}}(t_1^{\rho}, \dots, t_n^{\rho})$

Formulas are interpreted via the **satisfaction relation** $\mathcal{M} \models_{\rho} \phi$

(ϕ is true in $\mathcal{M}$ under environment ρ or. $\mathcal{M}$ satisfies ϕ under ...)

$\mathcal{M} \models_\rho P(t_1,\dots,t_n) \quad :\Leftrightarrow \quad P^{\mathcal{M}}(t_1^\rho,\dots,t_n^\rho)$

$\mathcal{M} \models_\rho t_1 = t_2 \quad :\Leftrightarrow \quad t_1^\rho = t_2^\rho$

$\mathcal{M} \not\models_\rho \bot$

$\mathcal{M} \models_\rho \neg\phi \quad :\Leftrightarrow \quad \mathcal{M} \not\models_\rho \phi$

$\mathcal{M} \models_\rho \phi \wedge \psi \quad :\Leftrightarrow \quad \mathcal{M} \models_\rho \phi$ and $\mathcal{M} \models_\rho \psi$ (podobno $\vee$ ... or)

$\mathcal{M} \models_\rho \phi \to \psi \quad :\Leftrightarrow \quad \mathcal{M} \not\models_\rho \phi$ or $\mathcal{M} \models_\rho \psi$

$\mathcal{M} \models_\rho \forall x\, \phi \quad :\Leftrightarrow$ for all $a \in M$, $\mathcal{M} \models_{\rho[x:=a]} \phi$; $\rho[x:=A] := z \mapsto \begin{cases} \rho(z) & ; \ z \text{ not } x \\ a & ; \ z = x \end{cases}$

$\mathcal{M} \models_\rho \exists x\, \phi \quad :\Leftrightarrow$ there exists $a \in M$ $\ni$: $\mathcal{M} \models \rho[x:=a]\, \phi$

If Φ is a sentence, then the truth value of $\mathcal{M} \models_\rho \phi$ is independent of ρ. We write $\mathcal{M} \models \Phi$.
A sentence Φ is **valid**, if for all structures $\mathcal{M}$, $\mathcal{M} \models \Phi$.
A sentence Φ is **satisfiable**, there exists a structure $\mathcal{M}$ $\ni$: $\mathcal{M} \models \Phi$.

EXAMPLE VALID FORMULAS

$[(\forall x.\ (\text{person}(x) \longrightarrow \text{mortal}(x)) \wedge \text{person}(\text{aristotle}) \longrightarrow \text{mortal}(\text{aristotle})]$

$\exists x.\ (\text{drinks}(x) \longrightarrow \forall y.\ \text{drinks}(y))$

EXAMPLE SATISFIABLE FORMULA : $\exists x \forall y.\ P(x,y)$

Satisfiability is undecidable for predicate logic:

THEOREM: There is no algorithm to solve the problem:
Church / Turing — input: a sentence of predicate logic
output: yes if valid
no if not valid

GÖDEL'S COMPLETNESS THEOREM:
The following are equivalent for a sentence Φ

1. Φ is provable in natural deduction
2. Φ is valid

$\Downarrow$ soundness
$\Uparrow$ completeness

The problem: inputs: a sentence Φ and a candidate proof of Φ
output: yes, if the candidate proof is indeed a correct proof of Φ
no, otherwise
is efficiently computable

The validity problem is not decidable
undecidable

THE BASIS OF PROOF ASSISTANTS (Lean, cog, agda)

The proof-checking problem is (efficiently) decidable.

The above, together with completeness gives us a proof-search algorithm for validity:
input: sentence Φ
algorithm: systematically search the space of all potencial proofs. Proof-check each one. If one is a proof then output the proof.

Theoretical result: Validity is semi decidable.

There is an algorithm that takes input: sentence Φ
output: yes, if Φ is valid
runs forever if Φ is not valid

Practical consequence of proof search: can refine using AI algorithms to improve the search ...

... THE BASIS OF THEOREM PROVERS

Satisfaibility is co-semi-decidable.

There is an algorithm that
- runs forever if Φ is satisfiable
- outputs no if Φ is not satisfiable

Bounded satisfiability

09 April 2024 08:30

SIGNATURE:
connected/2
server, client /1

AXIOMS: $\forall x. \neg connect(x,x)$
$\forall x \forall y\ connected(x,y) \rightarrow connected(y,x)$
$server(x) \vee client(x)$
$\neg server(x) \vee \neg client(x)$
$connected(x,y) \rightarrow \neg server(x) \vee \neg server(y)$
$\forall x.\ client(x) \rightarrow \exists y.\ server(y) \wedge connected(x,y)$
$\forall y.\ server(y) \rightarrow \exists x.\ client(x) \wedge connected(x,y)$
$\forall x \exists y.\ x \neq y \wedge \neg connected(x,y)$

Given a set S of sentences a MODEL S is a structure $\mathcal{M}$ s.t. $\forall \Phi \in S.\ \mathcal{M} \models \Phi$

ALGORITHMIC QUESTION: MODEL-CHECKING PROBLEM
Given a finite structure $\mathcal{M}$ and a formula Φ, is $\mathcal{M} \models \Phi$?

S-C-S ✓ $\overline{S-C-S}$ ✓

Algorithm for deciding the relation $\mathcal{M} \models_\rho \Phi$; $\mathcal{M}$ finite structure, Φ is a formula, $\rho : FV(\Phi) \rightarrow M$.
CASES:

$\mathcal{M} \models_\rho P(t_1, \dots, t_k)$: compute $t_1^\rho, \dots, t_k^\rho \in M$
check if $P^{\mathcal{M}}(t_1^\rho, \dots, t_k^\rho)$

$\mathcal{M} \models_\rho t_1 = t_2$: compute t_1^ρ, t_2^ρ
check if they are equal

$\mathcal{M} \models_\rho \neg\Phi$: return ¬(the result of $\mathcal{M} \models_\rho \Phi$)

$\mathcal{M} \models_\rho \Phi \wedge \Psi$: return $(\mathcal{M} \models_\rho \Phi)$ && $(\mathcal{M} \models_\rho \Psi)$

$\mathcal{M} \models_\rho \exists x.\ \Phi$: for $a \in M$
check $\mathcal{M} \models_{\rho[x:a]} \Phi$
return true iff at least one of above checks true

The above problem is decidable.
The algorithm works in time $O(|\Phi| \cdot |M|^d)$; d = #quantifiers

The model checking algorithm is not polynomial, but it is fixed-parameter tractable.
if we fix the formula Φ then the algorithm is polynomial in the structure size $|M|$.

Dodamo aksiom $\forall x \exists y.\ x \neq y \wedge \neg connected(x,y)$
Question: Does there exist a network that has only one server?
Is the formula $\forall x \forall y\ server(x) \wedge server(y) \rightarrow x = y$ satisfiable by a finite model? ZAPISKI.

FINITE SATISFIABILITY

Input: sentence Φ
Question: does Φ have a finite model? (ie. does there exist finite $\mathcal{M}$ st. $\mathcal{M} \models \Phi$?)

The above problem is semidecidable but not decidable.
Ialgorithm that outputs yes if Φ is finitely satisfiable and runs forever otherwise.

The above problem is semidecidable but not decidable.
∃algorithm that outputs yes if Φ is finitely satisfiable and runs forever otherwise.

Algorithm: systematically enumerate all finite structures $\mathcal{M}$ and for each check $\mathcal{M} \models \Phi$ using the model checking algorithm. Totally impractical!

BOUNDED SATISFIABILITY This is a more practical question.

Inputs: sentence Φ and a bound N
Question: does there exist an $\mathcal{M}$ st. $\mathcal{M} \models \Phi$ and $|M| \leq N$? and if so, output such a structure.
yes if ∃M and return $\mathcal{M}$, otherwise no.

A bounded model checker can tell us that there is no model with 1 server of size $\leq 7, 8, \ldots$

This problem is DECIDABLE, because there are only finitely many nonisomorphic structures of cardinality $\leq N$.

Algorithm for bounded satisfiability

We do this for Φ a formula in a signature with no function symbols and not containing equality.

Main idea: reduce to SAT (propositional satisfiability)

2 stages: (1.) predicate logic without (= and function symbols)
(2.) general predicate logic

STAGE 1:
Input formula Φ, bound N
Add N constants $c_1, \ldots, c_N$ to the signature.
Build propositional formulas using the following prop. variables
$P_{i_1,\ldots,i_k}$ for every predicate symbol P of arity k and $i_1, \ldots, i_k \in \{1, \ldots, N\}$.
idea: $P(c_{i_1}, \ldots, c_{i_n})$ is true
Define a mapping $\Phi \mapsto \Phi^*$, mapping predicate logic sentences Φ to prop. formulas Φ^*.

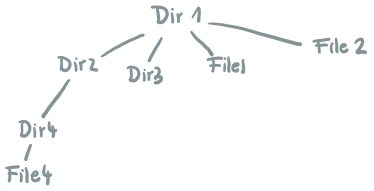
$(P(c_{i_1}, \ldots, c_{i_k}))^* = P_{i_1,\ldots,i_k}$
$(\neg\Phi)^* = \neg(\Phi^*)$
$(\Phi \wedge \Psi)^* = \Phi^* \wedge \Psi^*$
$(\exists x.\, \Phi)^* = (\Phi[c_1/x])^* \vee \ldots \vee (\Phi[c_N/x])^*$

Fact: Φ is satisfied by a structure $\mathcal{M}$ of cardinality $\leq N$ iff Φ^* is satisfiable.
If we have function symbols: replace every f of arity $n+1$
+ axioms: $\forall x_1, \ldots, x_n.\, \exists y.\, P_f(x_1, \ldots, x_n, y)$
$\forall x_1, \ldots, x_n, y, z.\, P_f(x_1, \ldots, x_n, y) \wedge P_f(x_1, \ldots, x_n, z) \rightarrow y = z$

and then we translate formulas involving f into equivalent formulas involving P_f.
Eg. $Q(f(x)) \rightarrow \exists y.\, Q(y) \wedge P_f(x, y)$.

Treat = as a binary predicate symbol by adding axioms
and for every predicate symbol P of arity n
$\forall x_1, \ldots, x_n, y_i.\, P(x_1, \ldots, x_n) \wedge x_i = y_i \rightarrow P(x_1, \ldots y_i, \ldots, x_n)$
$1 \leq i \leq n$
(n axioms)

$\forall x.\ x = x$
$x = y \rightarrow y = x$
$x = y \wedge y = z \Rightarrow x = z$



Alloy analyzer : a bounded satisfiability checker

Predicate: Parent/2 ... every object has at most one parent :
$\forall x,y,z.\ Parent(x,y) \wedge Parent(x,z) \longrightarrow y=z$

Predicates: Dir/1 , Contents/2 ... only directories have contents : $\forall x,y.\ contents(x,y) \longrightarrow dir(x)$

Predicates: File/1 , Root/1 ... $\exists x.\ root(x)$

$\forall x \forall y.\ root(x) \wedge root(y) \longrightarrow x=y$

$\forall x.\ root(x) \rightarrow \neg\exists y.\ parent(x,y)$

$\forall x.\ root(x) \rightarrow dir(x)$

$\forall x.\ file(x) \vee dir(x)$

$\forall x.\ \neg file(x) \vee \neg dir(x)$

$\forall x,y,z.\ dir(x) \wedge contents(x,y) \longrightarrow parent(y,x)$

$\forall x \exists y.\ root(y) \rightarrow \forall y.\ contents^{*}(x,y)$

Every x is REACHABLE from the root using the contents relation

$contents^{*}$: transitive reflexive closure

The alloy example uses the transitive-reflexive closure (in the reachability axiom) and the transitive closure (in the acycliaty property) of the contents relation.

In general: for a binary relation R, the TRANSITIVE CLOSURE R^{+} is: $xR^{+}y \iff xRy \vee \exists z_1.(xRz_1 \wedge z_1 Ry) \vee \vee \exists z_1 \exists z_2.(xRz_1 \wedge z_1 Rz_2 \wedge z_2 Ry) \vee \ldots$

The TRANSITIVE-REFLEXIVE CLOSURE: $xR^{*}y \iff xR^{+}y \vee x=y$.

The tr.cl. and tr-ref. cl. are not decidable in predicate logic. So the language of Alloy goes beyond predic. logic. Nonetheless bounded satisfiability for the language of Alloy is still reducible to SAT solving!